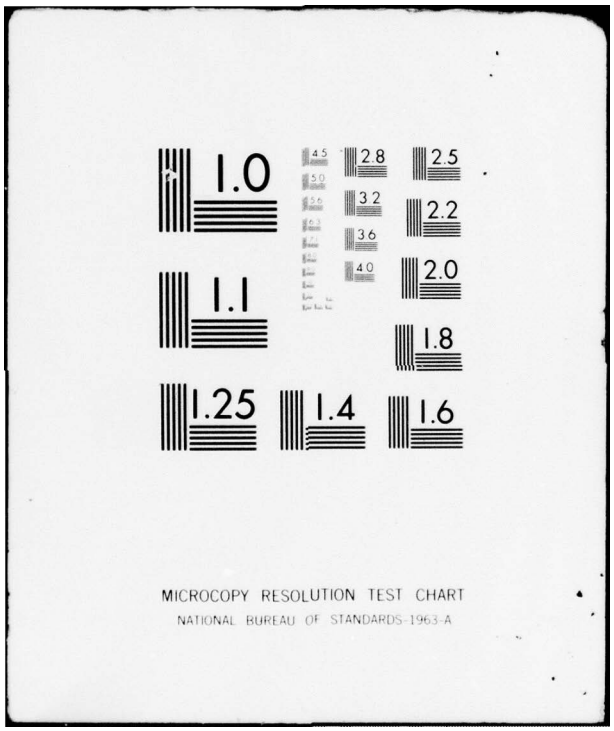
AD-A056 375 GENERAL ELECTRIC CO SCHENECTADY N Y RESEARCH AND DEV--ETC F/G 9/2
MADMAN ON THE H6000: DATA BASE MANAGER FOR HONEYWELL 6000 VOLUM--ETC(U)
JUN 78 D E PHILLIPS, J M BROWN, J P KELLERMAN F30602-76-C-0321

UNCLASSIFIED RADC-TR-78-8-VOL-1 NL

1 OF 1
AD A056375

END
DATE FILMED
9-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL II

(2)

RADC-TR-78-8, Volume I (of two)
Final Technical Report
June 1978

MADMAN ON THE H6000
Data Base Manager for Honeywell 6000

D. E. D. Phillips
J. M. Brown
J. P. Kellerman

General Electric Company

Approved for public release; distribution unlimited.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-78-8, Volume I (of two) has been reviewed and is approved for publication.

APPROVED: Donald Van Alstine

DONALD VAN ALSTINE
Project Engineer

APPROVED: Wendall C. Bauman

WENDALL C. BAUMAN, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISIM) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-78-8, Vol-I (of two)	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): MADMAN ON THE H6000 Data Base Manager for Honeywell 6000 Volume I.	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report 14 Jun 76 - 14 Jun 77
	6. PERFORMING ORG. REPORT NUMBER: N/A
7. AUTHOR(s): D. E. D. Phillips, J. M. Brown, J. P. Kellerman	8. CONTRACT OR GRANT NUMBER(s): F30602-76-C-0321
9. PERFORMING ORGANIZATION NAME AND ADDRESS: General Electric Company, Research and Development Center 1 River Road, Schenectady NY 12345	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F 55810269
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISIM), Griffiss AFB NY 13441	12. REPORT DATE: June 1978
	13. NUMBER OF PAGES: 21
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Donald Van Alstine (ISIM)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

MADMAN	CODASYL
Data Base Management	H6000
Command and Control	PDP-11
Distributed Systems	Network Data Base
Hierarchical	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

MADMAN is a multi-access data base management system that exists on both the PDP-11 and the H6000. This report describes the version that exists on the H6000 under GCOS Versions 1/G, 2/H and 3/I.

MADMAN is implemented in such a manner that only one copy of the data base manager is resident on the system for each data base that has been opened. That copy can support a maximum of 16 application programs at one time. The application programs are written in FORTRAN using the CALL statement to access the data base manager. Data is transferred core-to-core between the application

→ next page

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

programs and the data base manager.

Three key features of MADMAN are:

1. Complete implementation of the CODASYL network data base standard (except for SELECT UNIQUE MASTER).

2. Sophisticated concurrency control with automatic rollback in case of data base conflicts, and

3. Automatic rollbacks of processes that were active when systems crashed.

TABLE OF CONTENTS

EVALUATION

The objective of this effort was to implement an Information Management System Multi Access Data Management System (MADMAN) which until this effort was based solely on the PDP-11/45 Computer, on the Honeywell H6180 Computer System. This development falls within the goals of RADC TPO V, specifically in the 3.3 Tools and Procedures Area.

The General Electric Company, Corporate Research and Development Center, the Original Implementor of MADMAN on the PDP-11/40 and 45 computer systems, successfully re-implemented MADMAN on the H6180 computer system for RADC. An exhaustive test plan was successfully carried out to test all the capabilities of the system. The features tested fell into four parts:

1. Data Description Language and Data Manipulation Language Processors.
2. Concurrency Control
3. Database Utility Programs
4. Recovery from System Failure

RADC does not intend to do further work with this system at this time. However, many WWMCCS, Air Force and Air Force Intelligence users have been utilizing mini-computers to solve their data management and large data base file problems. In this application environment, and establishment or division of tasks of the data management system between the small and large machine is important. It is also important that the same data

management system be available on all computers in the systems. This simplifies data and application program interchange. In recognition of this concept MADMAN provides a software tool to meet the above requirements.

Donald Van Alstine

DONALD VANALSTINE
Project Engineer

1. INTRODUCTION TO MADMAN

This report is in two volumes. The first volume contains a description of MADMAN on the H6000 -- what it is and its capabilities. The second volume contains the manuals associated with MADMAN on the H6000.

1.1 Background

The Research and Development Center of the General Electric Company originally designed and implemented MADMAN on a PDP-11 for use in GE's internal manufacturing facilities. This version is presently being used in automation and information gathering capacities.

MADMAN was then moved to the H6000 under contract with Rome Air Development Center. Except for interfacing with the operating system, the design of MADMAN remained the same. The original implementation on the PDP-11 had been done using a set of macros and, except for the operating system interfaces, the reimplementation consisted of recoding the macro expansions. The detailed code logic flow and even the code itself are thus quite similar in the two implementations. The user FORTRAN interfaces are identical.

Because the H6000 implementation is, in fact, a reimplementation of a General Electric design, the design and coding documentation is released with limited and restricted rights as described in the contract.

1.2 The Data Base Administrator and System Administrator

In the description of how MADMAN is to be installed and maintained at any given site, we have assumed a particular model of how the system will be used and specifically we have assumed the existence of two individuals operating within the model: the Data Base Administrator and System Administrator.

For each application, the Data Base Administrator is responsible for designing the data base schema, encoding it in the data description language and providing each application programmer with the passwords and other information needed to write the application programs.

The System Administrator is responsible for creating and maintaining the various assemblies and files needed to establish a MADMAN system corresponding to a particular schema.

The Data Base Administrator needs to be knowledgeable in data base technology and the specific application. The System Administrator needs to be knowledgeable in the GCOS system.

1.3 Manuals

Throughout this report references are made to manuals written for the users of MADMAN on the H6000. The following list gives the name and a short description of each manual.

1) The Care and Feeding of MADMAN on the H6000

   This manual describes the step-by-step procedures which must be taken in order to install a version of MADMAN on any Honeywell H6000 series computer with GCOS Software Release 2/H or 3/I. It also includes a description of how to load MADMAN for the second, third or Nth data base. Control card set-ups for all steps are given. This manual is normally used by both the System Administrator and the Data Base Administrator.

2) Data Base Utility Manual

   This manual describes the format of all the commands available in both Data Base Utility One and Data Base Utility Two. It is normally used only by the Data Base Administrator.

3) Data Definition Language Manual

   This manual details the format for the description of the schema for a data base. It also is normally only used by the Data Base Administrator.

4) Data Manipulation Language Manual

   This manual describes the format of the commands to the Data Base Management System. It also describes the workings of a typical application program. This manual is the primary source of information for application programmers.

5) Prescan Manual

   This manual details the options of the command to the preprocessor routine that inserts the required FORTRAN declarations for the sets, records and items of a given data base into an application program. This manual is also for the use of application programmers.

2. OVERVIEW OF MADMAN

2.1 Parts of MADMAN DBMS

The acronym MADMAN stands for Multi-Access Data base MANagement system. MADMAN runs on the Honeywell 6000 series computer under the GCOS operating system, software releases 1/H, 2/G and 3/I. Its purpose is to provide efficient, on-line, concurrent access to network structured data stored on a disc by programs written in FORTRAN. There is one copy of the MADMAN Data Base Management System (DBMS) for each data base. Each copy is a free standing program that is spawned from the operator's console.

Each copy of MADMAN contains the following modules:

1. Dispatcher
2. Data Manipulation Language Processor (DMLP)
3. Page Manager (PM)
4. Data Base Utility One (DBU1)
5. Tables and Storage

2.1.1 Dispatcher

The dispatcher is responsible for communications between the application programs (AP) and the DBMS; and for driving the DMLP and the PM. Data is transferred between the AP's and the DBMS using the standard GCOS routine INTERCOM I/O which moves data from core in one program to core in another program. The section of the dispatcher using INTERCOM I/O essentially functions as a separate program. It responds to asynchronous interrupts caused by some application program's request. Then, depending upon the request, it allocates or de-allocates storage in the DBMS for the AP or it queues up a task for the DMLP. The dispatcher also responds to requests from the DMLP or PM to send data to the application program.

2.1.2 Data Manipulation Language Processor

The DMLP contains a set of routines corresponding to the verbs in the Data Manipulation Language. Each routine responds to some request made by the application program. Each checks that the request is valid and, if so, perform the request. If there is an error, the routine returns a specific error code.

In either case, the dispatcher returns the information to the application program using INTERCOM I/O.

2.1.3 Page Manager

The Page Manager is responsible for reading and writing pages of the data base from or to the disc. It is also responsible for concurrency control which is described in detail in section 3.3.

2.1.4 Data Base Utility One

There are two data base utility routines -- Data Base Utility One (DBU1) and Data Base Utility Two (DBU2). DBU2 is described in section 2.3.3. DBU1 is loaded as a part of the DBMS to permit the Data Base Administrator (DBA) to communicate with the data base before it becomes available to the application programs. DBU1 responds to the DBA's requests for such tasks as change DBMS parameters, initialize or verify the data base or put the data base on-line for the AP's to use. It also contains the routine that handles recovery. (See section 3.2 for details on the recovery procedure.) Once the DBU1 has completed its tasks, its core is returned to the system.

2.1.5 Tables and Storage

The tables and storage are assembled from files produced by the Data Definition Language (DDL) compiler. They tailor the DBMS to a specific data base. All the other parts of the DBMS are pure code and are only assembled once. The tables describe the data base to the rest of the DBMS and the storage initializes buffers for communication with the application program.

2.2 Application Programs

Each copy of MADMAN can communicate with sixteen concurrent FORTRAN application programs. From the FORTRAN programmers viewpoint, the communication consists of two parts:

1. A set of FORTRAN library routines that can be called from any FORTRAN program. The names of these routines, together with their arguments, comprise the Data Manipulation Language. The library routines, themselves, are referred to as the FORTRAN Language Interface or FLI.

2. FORTRAN labeled commons, one for each record, that contain the variables corresponding to the items in each record. The FLI routines put information into or take information out of these labeled commons, which are accessible to the FORTRAN program using standard FORTRAN statements. The declarations of these common areas are automatically inserted into each application program using the Prescan Routine described in Section 2.3.2.

When an application program wishes to use one of the DML verbs, it uses the FORTRAN CALL statement on the corresponding FLI routine. That routine does some preliminary checking on the validity of its arguments and then uses INTERCOM I/O to pass the specified information between a specific labeled common and the dispatcher and DBMS.

The verbs in the DML are similar to those proposed in the CODASYL COBOL Journal of Development, 1975. Several local verbs have also been added.

The verbs can be roughly divided in three groups. One group consists of the verbs used for data retrieval. For example FIND, FINDK, FINDD and FINDA locate a particular record depending on the verb and the arguments used. There are also four corresponding obtain (OBTN) verbs that locate the record and move it into a particular labeled common in the application program. Finally, there is a GET verb that moves some or all of the items of the current record into a particular labeled common in the application program.

A second group of verbs are used for data base updating. The verbs that are from CODASYL are STORE and STORED, MODIFY, INSERT, REMOVE and DELETE. Data base updating verbs that are not CODASYL are INCR, CHANGE, and CNGSET. INCR causes specified signed values to be added to given items in the current record. CHANGE assigns specified values to items in the current record. CNGSET reassigns the current record to specified sets.

The third group consists of miscellaneous verbs that open and close the data base, perform tests on set memberships and move currency table entries to the application program.

A set of reserved words exist that can be used as arguments in the calls to the DBMS. For example, FIRST can be used in FIND or OBTN to access the first record of a given set. RDONLY is used to indicate read-only access is requested on a data base open.

A complete description of all the verbs and reserved words is contained in the Data Manipulation Language manual. Examples of application programs are given in Appendix A.

Figure 1 shows an overview of the MADMAN Data Base Management System with application programs.

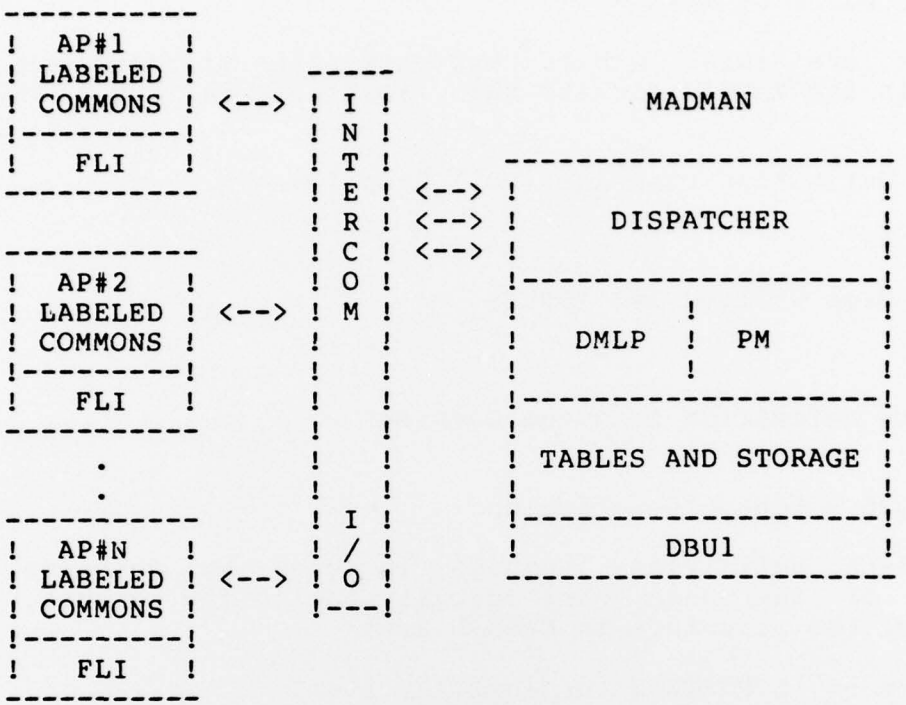

AP#1
LABELED
COMMONS
FLI
AP#2
LABELED
COMMONS
FLI
AP#N
LABELED
COMMONS
FLI
<-->
INTERCOM
I/O
MADMAN
DISPATCHER
DMLP
PM
TABLES AND STORAGE
DBU1

FIGURE 1

2.3 Other Parts of MADMAN

There are three other, separate, free standing programs supplied in the MADMAN package that facilitate the use of MADMAN DBMS.

1. Data Definition Language (DDL) Compiler
2. Prescan
3. Data Base Utility Two (DBU2)

2.3.1 Data Definition Language Compiler

2.3.1.1 DDL - Schema Description

The data definition language is used to describe the structure of the data base (usually called the schema). The elements of the structure in MADMAN are:

(1) Items -- In FORTRAN terminology, items can be integer, real, double precision, or character. One or more items make up a record. (In some terminology, items would be called the fields of the records).

(2) Records -- Records consist of groups of items that usually have some close correlation. One or more records make up a set.

(3) Sets -- Sets are groups of records that have some relationship to each other. Sets have an owner record and one or more member records. Records become members either automatically or manually. Set membership is optional, mandatory in a set, or fixed in a particular set. Dynamic sets are not supported; singular sets are.

(4) Areas -- Areas are a method of grouping records physically on some storage device whereas sets are a logical grouping.

2.3.1.2 Authority Key

The authority key feature of the DDL provides the Data Base Administrator (DBA) with the ability to limit the records and/or items to which an application program has access. The DBA

defines the authority keys (passwords) associated with particular groups of records or items. These keys cause PRESCAN to declare only those parts of the schema visable to the application program and cause the DBMS to access only the data visable to the program. Thus, the authority keys can be used to subset a schema.

2.3.1.3 DDL Compiler

The DDL compiler takes as input a file containing the schema description in the form shown in the Data Definition Language Manual. It produces files that are inputs to assemblies for the FORTRAN Language Interface (FLI), the Data Manipulation Language Processor (DMLP), and the Page Manager (PM). One file is input to Prescan. These assemblies are the only parts of MADMAN that change for each data base. The remainder of MADMAN is assembled only once.

2.3.2 Prescan

Prescan is a preprocessor for the FORTRAN application programs. Its input is the application program (AP) and a file describing the schema that was outputted from the DDL compiler. Within the application program is an INVOKE statement. It is a non-FORTRAN statement that is recognized by Prescan as containing the name of the schema and any restrictions that apply to this application program. Prescan converts the INVOKE statement to a comment and inserts the FORTRAN statements that describe the schema as this AP sees it. The statements consist of labeled common statements for each record and its items and PARAMETER statements for reserved words. DATA statements may also be included. After all the statements describing the schema are inserted into the AP, it is written into a file. This file is then compiled and run as an application program.

2.3.3 Data Base Utility Two

DBU2 is a free standing program written mostly in FORTRAN. Its purpose is to allow the Data Base Administrator to communicate with any DBMS while the data base is on-line. When requested, DBU2 gives such information as what AP's are running against the data base and what the DBMS parameters are. It can also be used to generate statistics and to shut the data base down. See the Data Base Utility manual for the complete list of commands.

3. DATA BASE INTEGRITY

This section discusses three additional features of the MADMAN system that affect the integrity of a data base.

1. Rollback
2. Recovery
3. Concurrency

3.1 Rollback

When an application program makes any change to the data base that causes the DMLP to write out a new version of a data base page, the Page Manager copies the old version of that page to a "before" file. If later that program terminates normally, its before file space is released and its modifications to the data base become permanent. If, however, the AP aborts, the PM reads all the pages associated with that AP from the before file and writes them back into the data base, thereby removing the changes made by the application program.

This process of restoring the modified pages is called rollback. One of its uses is to insure that, if an application program does not terminate normally, its modifications to the data base are removed.

3.2 Recovery

Another use of rollback is in data base recovery in case of system crashes. Recovery is a procedure that is automatically invoked when DBUl discovers upon receiving the command from the DBA to put the data base on-line, that the data base was not shut down correctly the last time it was on-line. This situation can occur when GCOS crashes or the DBMS is aborted. The triggering mechanism is the contents of the before file.

When the Data Base Administrator requests that a data base be put on-line, DBUl checks the before file. If the DBMS was not shut down normally, recovery causes the contents of the before file to be written to the appropriate data base pages. Thus, any modifications that were made by application programs running at the time of the crash are removed.

3.3 Concurrency Control

3.3.1 Description

The concurrency control is that portion of the Page Manager that is concerned with deciding what actions should be taken in response to requests by individual application programs to read or write into the data base. The MADMAN concurrency control is at the system level. Individual application programs do not contain statements to lock and unlock data base entities they access, and, in fact, each application program is written as if it were the only program running on the system.

The concurrency control is concerned with avoiding deadlocks or similar occurrences that can prevent program termination and with maintaining the consistency of the data base.

The concurrency control operates on the data base page level. When one application program requests access to a data base entity (item, record, etc), the concurrency control places whatever locks are necessary on the entire page on which that entity resides.

The concurrency control allows multiple readers or a single writer to access any one page concurrently. Thus, if a program has had a read request for an entity on a given page granted, any other program can read that page but if another program requests to write onto that page, that program is made to wait until all of the reading programs have terminated, aborted or cleanpointed (see below), thus releasing the lock on the page. Similarly, if one program has had a write request for a page granted, any other programs requesting to read or write on that page will be made to wait until the lock on that page is released.

If a set of programs that have been made to wait should happen to form a cycle of programs waiting for each other, a deadlock is said to have occurred. The concurrency control detects this situation and breaks the deadlock by aborting one or more programs and rolling back their changes using the before file.

All locks the concurrency control has placed on data base pages on behalf of an application program are released when the program terminates, aborts or cleanpoints.

3.3.2 Terminate

DBTERM is a verb in the Data Manipulation Language that causes the data base to be closed to the application program. It also has an argument that permits the modifications to be made permanent or to be rolled back.

3.3.3 Abort

When an application program aborts, all its modification to the data base are rolled back automatically.

3.3.4 Cleanpoint

DBCLN is a verb in the DML that permits the application program to declare a cleanpoint. A cleanpoint causes a close and an open on the data base with one access. An argument permits the close to be a normal terminate, in which case the modifications done by the AP up to this point in the program become permanent, or a rollback, in which chase the Page Manager removes the modifications to the data base. In either case, an open is done immediately after the close and the application program is considered a new program.

3.3.5 Design

The design of the MADMAN concurrency control is based on the paper Concurrency Control for Database Systems [1] and can be proved to work correctly in the sense that, if each individual program when supplied with a consistent data base will eventually terminate without destroying the consistency of the data base, then during the concurrency operation of any set of application programs,

1. Each program sees a consistent data base
2. Each program eventually terminates
3. The final data base after all processes terminate is consistent.

REFERENCES

1. R.E. Stearns, P.M. lewis II and D.J. Rosenkrantz
Concurrency Control for Database Systems

2. CODASYL Data Description Language Journal
of Development. June 1973. National Bureau of Standards.
NBS Handbook 113.

3. CODASYL COBOL Journal of Development - 1975.

APPENDIX A

EXAMPLES OF APPLICATION PROGRAMS

BEFORE PRESCAN

```
C      THIS PROGRAM UPDATES THE RECEIT FIELD OF THE SUPPLY
C      RECORD AND SIGNALS THE SUPPLY ORDER AS COMPLETED WHEN
C      THE FULL SUPPLY ORDER IS FILLED. THIS IS DONE BY PLACING
C      THE SUPPLY ORDER IN THE SUPP SET OWNED BY THE 'C' STATRC.
C
C      R. D. LORDI                    JUNE 1976
C
       INVOKE SCHEMA DEMDB
       CHARACTER HOLD*80,PAYNO*10
       INTEGER STAT,ORDER,RCV
       READ(5,999) PAYNO
  999  FORMAT(A10)
       STAT=0
       CALL MSGT(9,'FDBACK IN',6,$1000)
C
       CALL DBOPEN(DEMDB,-1,0,0,32,ERL,$1001)
       CALL PROMPT(18,'ENTER LOCATION ID#',HOLD,5,$1000,$1000)
       CALL PICK(HOLD,5,ADCIDN)
  900  FORMAT(I5)
       CALL FINDK(ADCTRC,ANY,ADCIDN,ERL,$1002)
       CALL FIND(NEXT,ADCTST,STATRC,ERL,$1003)
       CALL OBTN(NEXT,ADCTST,STATRC,ERL,$1003)
       IF(ACSTAT.NE.'R') GO TO 1003
       CALL PROMPT(18,'ENTER SUPPLY ORDER#',HOLD,5,$1000,$1000)
       DECODE(HOLD,900,ERR=1000) ORDER
     5 CALL OBTN(NEXT,SUPP,SUPPLY,ERL,$1004)
       IF(ORDNO.NE.ORDER) GO TO 5
       CALL PROMPT(18,'ENTER QTY RECEIVED',HOLD,5,$1000,$1000)
       DECODE(HOLD,900,ERR=1000) RCV
       CALL INCR(RECEIT,RCV,ERL,$1005)
       CALL GET(RECEIT,ERL,$1005)
       IF(RECEIT.LT.SUPQTY) GO TO 30
       CALL OBTN(NEXT,ADCTST,STATRC,ERL,$1003)
       IF(ACSTAT.NE.'C') GO TO 1003
       CALL FIND(CURRNT,NULL,SUPPLY,SETS,ERL,$1003)
       CALL CNGSET(SUPP,ERL,$1003)
       WRITE(6,901,ERR=1000) ORDER
  901  FORMAT(/,' SUPPLY ORDER #',I5,' COMPLETED')
       GO TO 30
C
 1000  CALL MSGB(8,'IO ERROR',6,$40)
 1001  CALL MSGB(12,'DBOPEN ERROR',6,$45)
 1002  CALL MSGB(16,'INVALID ADC NAME',6,$40)
 1003  CALL MSGB(22,'STATRC RETRIEVAL ERROR',6,$45)
 1004  CALL MSGB(19,'INVALID SUPPLY ORDER',6,$40)
 1005  CALL MSGB(10,'INCR ERROR',6,$45)
 1006  CALL MSGB(18,'SUPPLY STORE ERROR',6,$45)
   45  CALL MSG(17,'INFORM SUPERVISOR',6,$40)
   40  STAT=1
   30  CALL DBEND(5,6,ERROR,STAT,$1000)
       STOP
       END
```

```
C
C       THIS PROGRAM SEEKS ALL SUPPLY RECORDS BEARING A STRTDT
C       LESS THAN OR EQUAL TO AN INPUTTED DATA AND OWNED BY
C       A STATRC RECORD OF STATUS TYPE 'P', AND MOVES THEM TO
C       THE ORDINATE SUPP SET WHOSE OWNER IS OF STATUS TYPE 'R'.
C       THIS 'RELEASES' A SUPPLY ORDER. ALL RELEASES PRINTED.
C
C       R. D. LORDI                     JUNE 1976
C
        INVOKE SCHEMA DEMDB
        CHARACTER HOLD*80,ENDSET*1/025/,PAYNO*10
        INTEGER STAT,DAY
        REAL OHD,STRKEY,SUPKEY(4)
        READ(5,999) PAYNO
  999   FORMAT(A10)
        STAT=0
        CALL MSGT(9,'ORDREL IN',6,$1000)
        CALL ATTACH(11,'M6000/DEMO/DUMP104;',3,0,ISTAT, )
        CALL MSGT(22,'RELEASED SUPPLY ORDERS',11,$1000)
C
        CALL DBOPEN(DEMDB,-1,0,0,32,ERL,$1001)
        CALL PROMPT(19,'ENTER MAX STRTDT(5):',HOLD,5,$1000,$1000)
        DECODE(HOLD,900,ERR=1000) DAY
  900   FORMAT(I5)
        WRITE(11,901,ERR=1000)
  901   FORMAT(/,' ORD#',3X,'FROM',4X,'TO',9X,'MATERIAL',17X,
&              'DESCRIPTION',12X,'QTY',3X,'START',3X,'DUE',5X,
&              'COST',/,1X,105('-'))
    3   CALL OBTN(NEXT,ADCLST,ADCTRC,ERL,$200)
        CALL OBTN(NEXT,ADCTST,STATRC,ERL,$1003)
        IF(ACSTAT.NE.'P') GO TO 1003
        CALL MOVEC(STRKEY)
        CALL OBTN(NEXT,SUPP,SUPPLY,ERL,$100)
        CALL OBTN(NEXT,ADCTST,STATRC,SUPP,ERL,$1003)
        IF(ACSTAT.NE.'R') GO TO 1005
    6   CALL MOVET(SUPP,SUPKEY)
        IF(STRTDT.GT.DAY) GO TO 5
        CALL OBTN(OWNER,MATSPC,MATITM,ERL,$1004)
        OHD=ICS*SUPQTY
        WRITE(11,902,ERR=1000) ORDNO,DSTADC,ADCIDN,MATIDN,DESCRI,
&                              SUPQTY,STRTDT,SUPDUE,OHD
  902   FORMAT(1X,I5,2X,A5,2X,A5,2X,A18,2X,A30,3(2X,I5),2X,F10.4)
        CALL OBTN(CURRNT,NULL,STATRC,ERL,$1003)
        CALL FIND(CURRNT,NULL,SUPPLY,SETS,ERL,$1006)
        CALL CNGSET(SUPP,ERL,$1007)
    5   IF(SUPKEY(2).EQ.STRKEY) GO TO 3
        CALL OBTND(SUPKEY(2))
        GO TO 6
  100   IF(ERROR.EQ.ENDSET) GO TO 3
        GO TO 1008
  200   IF(ERROR.EQ.ENDSET) GO TO 20
        GO TO 1009
```

```
1000 CALL MSGB(8,'IO ERROR',6,$30)
1001 CALL MSGB(12,'DBOPEN ERROR',6,$30)
1003 CALL MSGB(16,'NO P STAT RECORD',6,$30)
1004 CALL MSGB(29,'MATSPC/MATITM RETRIEVAL ERROR',6,$30)
1005 CALL MSGB(16,'NO R STAT RECORD',6,$30)
1006 CALL MSGB(29,'CURRNT SUPPLY RETRIEVAL ERROR',6,$30)
1007 CALL MSGB(12,'CNGSET ERROR',6,$30)
1008 CALL MSGB(27,'SUPP/SUPPLY RETRIEVAL ERROR',6,$30)
1009 CALL MSGB(29,'ADCLST/ADCTRC RETRIEVAL ERROR',6,$30)
  30 STAT=1
  20 CALL DBEND(5,11,ERROR,STAT,$1000)
     STOP
     END
```